Name: Maura Kieft
ID: 103947905

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 11 **Spring 2019, CU-Boulder**

A startup has hired you as the chief technology officer (i.e., the only one who knows how to program). After the founders (all MBAs) finish explaining their vision for changing the world, you realize that what they describe can be reduced to the traveling salesman problem. No worries, you develop a solution that is a 1.5 approximation. The founders are devastated that they cannot use the word "optimal" in their next VC pitch, and wonder out loud if they need to get a new CTO who can do better. Convince them that an efficient optimal solution is unlikely (i.e., P probably does not equal NP) and that your solution is quite good.

Since the founders vision to change the world can be reduced to the traveling salesman problem, a famous NP problem, there is no "optimal" known solution for this problem. The ability to even develop a solution that is a 1.5 approximation to this problem is actually quite good, and we most likely would not be able to find a better solution. In fact, an efficient optimal solution is very unlikely. To explain why this is true, I wish to first define a few terms for you. First, the word "optimal" in regards to solutions. In the computer science world, an algorithm is said to be "optimal" roughly speaking if, for large inputs, it performs at worst a constant factor, which is independent of the input size, worse than the best possible algorithm. In order to explain why the solution, as a 1.5 approximation, is actually good, I must also explain what P and NP entails. So, P is a set of problems that can be solved by a deterministic Turing machine in polynomial time. A Turing machine is a theoretical machine that is used in thought experiments in order to examine the limitations and abilities of algorithms. A deterministic Turing machine has a set of rules which imposes, at most, one action to be performed for any given situation. NP is a set of decision problems, like the traveling salesman problem, in which NP stands for non-deterministic polynomial time. It basically means that the problem can be solved in polynomial time using a non-deterministic Turing machine.A non-deterministic Turing machine has a set of rules that requires more than one action for any given situation. Essentially, a solution has to be feasible in poly time. A problem is NP-hard, a subset of NP, if every problem in NP is reducible to L in polynomial time. NP-complete problems, which are a subset of NP-hard problems, is a known NP problem that can be solved using the problem with modified input, or, in other words, the NP problem can be reduced to the given problem. In our case, our plan for changing the world is reduced the traveling salesman problem. The main takeaway from an NP-complete problem, like our vision for changing the world, is that it **cannot** be solved in polynomial time (its not "optimal") in **any** known way. NP-hard and NP-complete shows that certain classes of problems are not solvable in realistic time. No polynomial time algorithm has yet to be discovered for any NP-complete problem, and, because of this, we find approximation solutions which usually tell us how close the approximation is. In this case, my solution is a 1.5 approximation. In order to explain this, I'll first explain the traveling salesman problem.

Name: Maura Kieft
ID: 103947905

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 11 **Spring 2019, CU-Boulder**

The traveling salesman problem is, given a set of cities and distance between every pair of cities, the problem is to find the hamiltonian cycle(or tour) of the cities with minimum cost. So, given a complete undirected graph $G = (V, E)$ that has nonnegative integer cost $c(u, v)$ associated with each edge (u, v). Since the problem is NP-complete, it is unlikely that we can find a polynomial-time (or "optimal") algorithm for the traveling salesman problem. So, in order for there to be a polynomial-time algorithm, P = NP. In order to prove that $P \neq NP$ and that there is no polynomial-time("optimal")solution to our problem, suppose that $c(A)$ denotes the total cost of the edges in the subset A subset of E;

$$c(a) = \Sigma_{u,vinA} c(u, v)$$

If we also suppose that there exists a polynomial-time algorithm for A, we use this algorithm to solve an instance of the hamilton cycle/tour in polynomial time. Since the hamilton problem is NP-complete, it leads us to the theorem:

If any NP-complete problem is polynomial-time solvable, then P=NP. Equivalently, if any problem in NP is not polynomial-time solvable than no NP-complete problem is polynomial solvable.

In order to solve the hamilton problem in polynomial time, it implies that $P = NP$. But, since the hamilton-tour is a subset of the traveling salesman problem, it is NP-complete, and therefore, $P \neq NP$.

Thus, we can use an approximation algorithm to solve the problem since there is no optimal/polynomial-time solution. An approximate algorithms work only if the problem instance satisfies something called the Triangle-Inequality. The Triangle-Inequality, states:

The least distant path to reach a vertex j from i is always to reach j directly from i, rather than through some other vertex(or vertices) k.
Basically dis(i,j) is always, $dis(i, j) \leq dis(i, k) + dis(k, j)$.

So, since the cost function of the traveling salesman problem satisfies the triangle inequality, we can design an approximate algorithm for the traveling salesman problem which returns a tour whose cost is never more than twice the cost of an optimal tour. In our case, I developed a solution that is a 1.5 approximation which essentially means that I could only find a tour

from the hamiltonian cycle that is 1.5 times more than the "optimal" tour. Although I could not find a smaller tour, what I did find is not too large either, and will probably be the best, efficient solution. So, you can find a new CTO, but they will most likely not do better than my solution unless they are about to solve every single NP problem. In that case, you would have to find another CTO since that is basically Nobel Prize winner material.

Name: Maura Kieft
ID: 103947905

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 11 **Spring 2019, CU-Boulder**